Robust Image Segmentation in Low Depth Of Field Images

Franz Graf, Hans-Peter Kriegel, and Michael Weiler

Ludwig-Maximilians-Universitaet Muenchen,
Oettingenstr. 67, 80538 Munich, Germany
[graf,kriegel,weiler]@dbs.ifi.lmu.de

February 19, 2013

Abstract

In photography, low depth of field (DOF) is an important technique to emphasize the object of interest (OOI) within an image. Thus, low DOF images are widely used in the application area of macro, portrait or sports photography. When viewing a low DOF image, the viewer implicitly concentrates on the regions that are sharper regions of the image and thus segments the image into regions of interest and non regions of interest which has a major impact on the perception of the image. Thus, a robust algorithm for the fully automatic detection of the OOI in low DOF images provides valuable information for subsequent image processing and image retrieval. In this paper we propose a robust and parameterless algorithm for the fully automatic segmentation of low DOF images. We compare our method with three similar methods and show the superior robustness even though our algorithm does not require any parameters to be set by hand. The experiments are conducted on a real world data set with high and low DOF images.

1 Introduction

In photography, low depth of field (DOF) is an important technique to emphasize the object of interest (OOI) within an image. Low DOF images are usually characterized by a certain region which is displayed very sharp like the face of a person and blurry image regions which are significantly before of behind the object of interest.

Low DOF images are well known from sports, portrait or macro photography where only a specific part of the image should attract most of the users' attention. The OOI is thereby displayed sharp while other areas like the background appears blurred, so that the viewer automatically focuses on the sharp areas of the image. When viewing a low depth of field image, the viewer implicitly segments the image into regions of interest and regions of less interest (usually background). As this implicit segmentation has major impact on the perception of the image, this information is a valuable feature for the subsequent image processing chain like an adaptive image compression [6] or image retrieval aspects such as the similarity of images which can be considerably influenced by the image's DOF. Given for example two images displaying a person in the sharp image region in front of different, blurred backgrounds, people might judge both pictures similar even though the blurred background differs. Although this implicit segmentation is rather easy for a human viewer of the photo, it is not an easy task for a completely unsupervised algorithm. This can be explained by the fact that there is usually not a sharp edge which divides the sharp OOI and blurred background. Depending on the camera's setting, this transition can be very smooth so that it is hard to distinguish where the OOI ends.

With the vastly growing market of consumer

DSLRs or even new small compact cameras like the Sony Cybershot which are explicitly being advertised with the ability for low DOF photos, the amount of low DOF photos also increased. This growing amount of low DOF images may also provide new information for established search and retrieval systems if they take the OOI into account when performing the similarity search tasks. In order to profit from the low DOF information, search engines and feature extraction algorithms need fully automatic and robust image segmentation algorithms which can separate the OOI from the rest of the image. For large search engines or image stock agencies, such algorithms should also be independent of the image domain, image size and the color depth of the image so that the algorithm performs well, no matter of the color or the toning of the photo (e.g. black-and-white, color photos, sepia photos).

In this paper, we propose a robust, fully automatic and parameterless algorithm for the segmentation of low DOF images as well as an analysis of the impact of low DOF information on similarity search. The algorithm does not need any a priory knowledge like image domain or camera settings. The algorithm also provides meaningfull results even if the DOF is rather large so that the background provides significant structures. The rest of the paper is organized as follows: In Sec. 2 we review some related work and some technical background, follwed by the explanation of the algorithm in Sec. 3. In Sec. 4 we explain our experimental evaluation of the algorithm. In Sec. 5 we describe the *internal* parameter settings and threshold values. The impact of the DOF segmentation to image similarity is shown in Sec. 6. Afterwards we finish the paper with a conclusion and outlook in Sec. 7.

2 Related work

The segmentation of low DOF images has gained some interest in the research community in past years. In [12, 14] early approaches to segment low DOF images were presented. Thereby [14] is using an edge-based algorithm which first converts a gray-scale image into an edge-representation which is then filtered. Afterwards the edges are linked to form closed boundaries. These boundaries are treated with a region filling process, generating the final result. [12] presents a fully automatic segmentation algorithm using block based multiresolution wavelet transformations on gray scale images. Even though the paper lists high rates of sensitivity and specificity on the testset, the authors also name some limitations like the dependence on very low DOF, fully focused OOI, and high image resolution and quality. In [16, 15] high frequency wavelets are used to determine the segmentation of low DOF images. As stated in [9], these features have the drawback of being not too robust if used alone and thus often result in errors in both focused and defocused regions if the defocused background shows some busy textures or if the focused foreground does not have very significant textures. In [10], localized blind deconvolution is proposed to determine the focus map of an image. Yet the authors do not propose a pure image segmentation algorithm as the focus map is not a true segmentation but a measure for the amount of focus in this part of the image. Also the algorithm does not take into account any color information as it is only operating on gray scale images. The works proposed in [9, 13, 7, 8] are consecutive works for segmentation of DOF images and sequences of images [9, 13] like in movies which address a similar topic. In this paper, we were inspired by the algorithm proposed in [7] which uses morphological filtering for the segmentation. Some problems of this algorithm were given by background that showed significant structures or if a photo was taken with high ISO values. Also the algorithm showed some problems if spatially separated OOIs were shown in a single image. Another problem can be raised by the size of the structuring element used in the algorithm [7, Sec. IV].

We compare our algorithm with the work of [7], with [11] where single frames of videos are processed into a saliency map which is processed by morphological filters. The resulting tri-map is then used for error control and for the extraction of boundaries of the focused regions. We also compare our work to the algorithm proposed in [18], where a fuzzy segmentation approach was proposed by first separating the image into regions using a mean shift. These regions

are then characterized by color features and wavelet modulus maxima edge point densities. Finally, the region of interest and the background are separated by defuzzification on fuzzy sets generated in the previous step. Our test image dataset consists of a set of various photos and comprises several categories from high to low DOF images.

2.1 Depth of Field

In optics, the DOF denotes the depth of the sharp area around the focal point of a lens seen from the photographer. Technically, each lens can only focus at a certain distance at a time. This distance builds the focal plane which is orthogonal to the photographers view through the lens. Precisely, only objects directly on the focal plane are absolutely sharp, while objects before or behind the focal plane are displayed unsharp. With increasing distance from the focal plane, the sharpness of the displayed object decreases. Nevertheless, there is a certain range before and behind the focal plane where objects are recognized as sharp until a blur is perceived. The depth of this region is then called the DOF. As the sharpness decreases gradually with increasing distance from the focal plane, it is hard to determine an exact range for the DOF as the limits of the sharp area are only defined by the perceived sharpness.

Points in the defocused areas appear blurred to a certain degree. This is often modeled by a Gaussian kernel G_σ as in Eq. 1 where σ denotes the spread parameter which affects the strength of the blur. For a given image I, the blurred representation can then be created by a convolution $G_\sigma * I$.

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \qquad (1)$$

The effect of DOF is mainly determined by the choice of the camera respectively its imaging sensor size, aperture and distance to the focussed object. The larger the sensor or aperture, the smaller the DOF. Increasing the distance from the camera to the focussed object will also expand the resulting DOF. Figure 1 illustrates the geometry of DOF at a symmetrical lens.

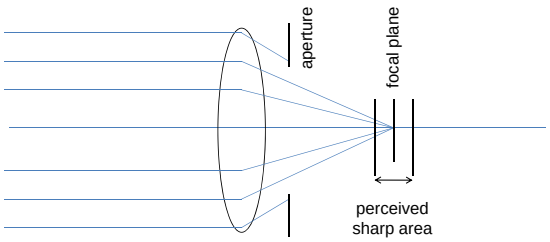


Fig. 1: Figure illustrating the depth of field. The size of the perceived sharp area around the focal plane denotes the DOF.

2.2 Automatic segmentation of low DOF images

Automatic segmentation of images is more challenging than interactive approaches because no additional information of humans can be used to adapt parameter values for the segmentation process. However the advantages of a fully automated algorithm are obvious, if the according algorithm should be deployed to a system providing lots of images where the segmentation should be present as fast as possible. This is for example the case in search index or photo communities like Flickr or Google's Picasa, where several thousand photos are uploaded each minute, even if not all of them are low DOF images.

The requirements to a segmentation algorithm are that it should be able to handle different types (grayscale or color), orientations (landscape or portrait) and resolutions (from small to large) of images, independent of the camera settings like ISO etc. Many automatic segmentation approaches of low DOF images have some of these restrictions, as seen in [18], which only performs well on color images. Grayscale images mostly fail because the extracted color features are too few, to characterize regions and distinguish them sufficient.

However, other algorithms like the one presented in [7], can only process grayscale images. In such cases, color images have to be transformed and hence their color information looses its contribution to improve segmentation quality. As shown in our experimental

results in section 4, images that consist of complex defocused regions can cause poor segmentation results, because too many false positives are found. In this context, false positives describe the set of pixels that are defined as background by the underlying ground truth but classified as OOI pixel by the segmentation algorithm.

In the following section, we describe our algorithm that does not suffer from one of the restrictions mentioned above. Therefore, we use a robust method for calculating the amount of sharpness of a pixel in relation to its neighbors by taking advantage of the $L^*a^*b^*$ color space, which offers a more accurate matching between numerical and visual perception differences between colors. The $L^*a^*b^*$ model was favored over the well known RGB and $CMYK$ color spaces, as the $L^*a^*b^*$ model is designed to approximate human vision better than the other color spaces.

To accomplish the problems caused by images consisting of numerous less blurred pixel regions showing complex structures, we apply a density-based clustering algorithm to all found sharp pixels. This enables our algorithm to distinguish between sharp pixels belonging to the main focus region of the OOI (if these pixels belong to the largest found cluster) and noise pixels located in background structures.

3 Algorithm

The proposed algorithm consists of the following five stages: *Deviation Scoring*, *Score Clustering*, *Mask Approximation*, *Color Segmentation* and *Region Scoring*. Before explaining the steps of the algorithm in detail, we first want to give a brief summary of the complete algorithm. Fig. 2 illustrates the steps of the algorithm.

The first stage of the algorithm, called *Deviation Scoring*, identifies sharp pixel areas in the image. Therefore a Gaussian Blur is applied to the original image. The difference between the extracted edges from the original image and the blurred image is then calculated. For each pixel, this difference represents a score value, with higher score values indicating sharper pixels and lower score values indicating blurred pixels.

In the second stage, called *Score Clustering*, all pixels with a score value above a certain threshold are clustered by using a density-based clustering algorithm. Thus, isolated sharp pixels are recognized as noise and only large clusters are processed further.

The third stage named *Mask Approximation* generates a nearly closed plane (containing almost no holes) from the discrete points of each remaining cluster. This is achieved by computing the convex hull from all neighbors of all dense pixels. Any so-created polygon is then filled and the union of these filled regions represent the approximate mask of the main focus region. In the next two stages this approximate mask is going to be refined.

Hence, the fourth stage, called *Color Segmentation* divides the approximate mask into regions that contain pixels with similar color in the original image.

In the fifth stage named *Region Scoring*, a relevance value is calculated for each region. This relevance value is directly influenced by the score values of the pixels surrounding the according region. The final segmentation mask is then created by removing all regions that have a relevance value below a certain threshold.

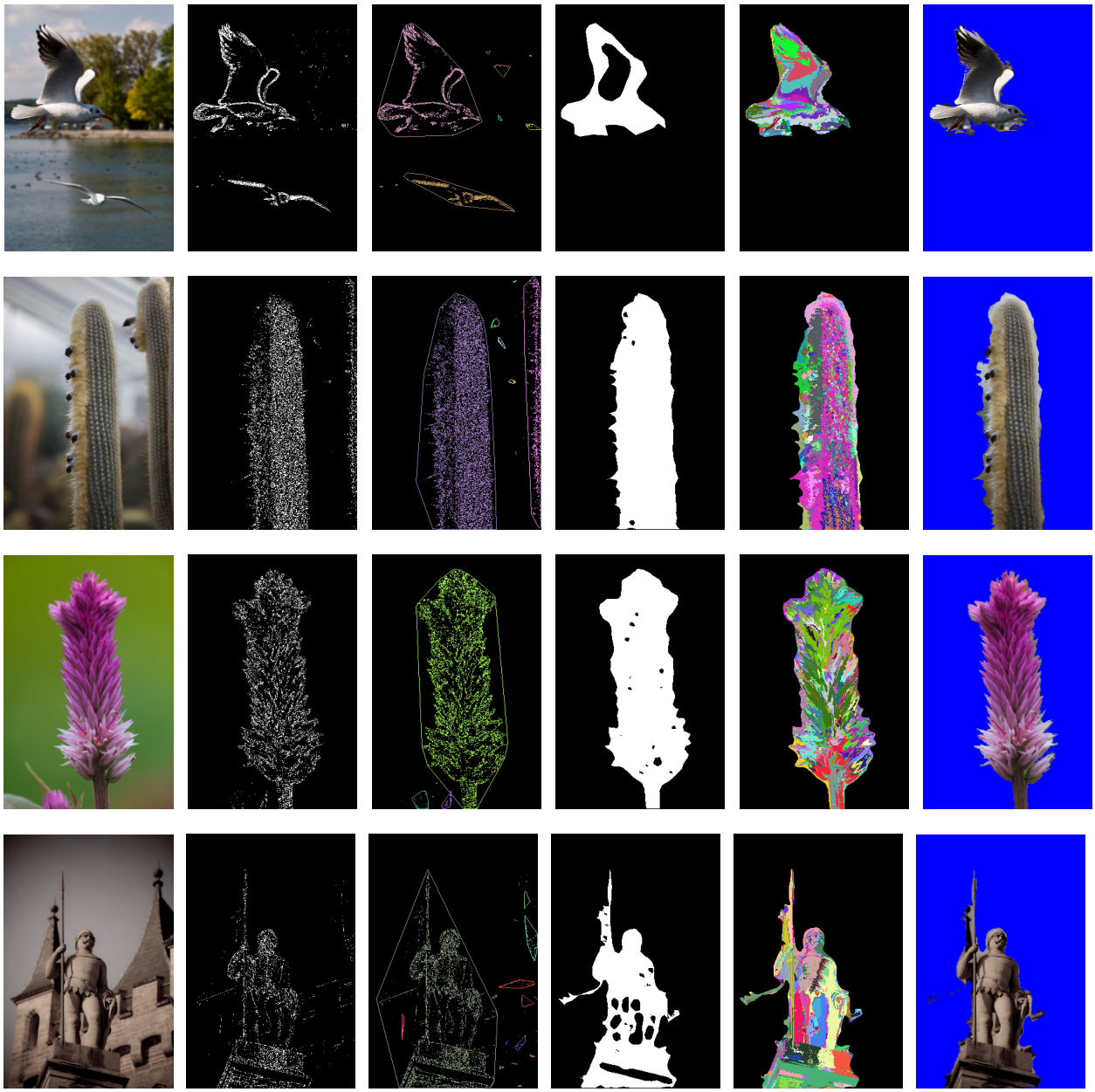
3.1 Deviation Scoring

In the first step, we need to identify sharp pixels as an indication for the focused objects within the image. The well known Canny edge detector [2] is not suitable in this case because the Canny detector operates on gray scale images and not on the $L^*a^*b^*$ color space. Furthermore, the Canny operator does not aim at the detection of single edge pixels but at the robust detection of lines of edges even in partly blurred areas of the image (c.f. Fig. 3b on page 6).

The HOS map used in [7] is defined as in Eq. 2

$$HOS\left(x,y\right) = min\left(255, \frac{\hat{m}^{(4)}\left(x,y\right)}{DSF}\right), \qquad (2)$$

where DSF represents a down scaling factor of 100 and the forth-order moment $\hat{m}^{(4)}$ at (x,y) is given by $\hat{m}^{(4)}\left(x,y\right) = \frac{1}{N_\eta} \sum_{(s,t)\in\eta(x,y)} \left(I\left(s,t\right) - \hat{m}\left(x,y\right)\right)^4$ where



(a) Input Image (b) Deviation Scoring (c) Score Clustering (d) Mask Approximation (e) Color Segmentation (f) Region Scoring

Fig. 2: Illustration of the five stages of our algorithm: Fig. 2a: Input image with low DOF and relatively complex background regions. Fig. 2b: Identify sharp pixels by computing the difference between the edges of the original image and the edges of the blurred version of the image. Fig. 2c: Generate clusters from pixels with a high appropriate score by a density-based clustering algorithm (for a better visual representation we colored each found cluster and surround it with its convex hull). Fig. 2d: Filling all convex hulls from all neighbors of all dense pixels. Fig. 2e: Group pixels into regions that contain similar colored pixels in the original image. (For a better visual representation we colored each found region in a random color). Fig. 2f: Removing all color Regions with low relevancy.

$\hat{m}$ is the sample mean and defined as in Eq. 3

$$\hat{m}(x,y) = \frac{1}{N_\eta} \sum_{(s,t)\in\eta(x,y)} I(s,t). \tag{3}$$

Thereby $\eta(x,y)$ is the set of neighborhood pixels with center (x,y) and is set to size 3×3 where N_η denotes its cardinality. Using the HOS map also has the disadvantage that it operates only on gray scale images. Additionally, the HOS map is too sensitive in case of textured background as it only produces reasonable results if the background is significantly blurred. This works for images with very low DOF, but as soon as the DOF is not very small, the HOS map detects too many sharp areas in the background (c.f. Fig. 3c).

Thus, we propose the process of Deviation Scoring. Let I be the set of pixels of the processed image. For each pixel $p(x,y) \in I$, the mean color from the pixel's r-neighborhood is calculated by

$$\eta^r_{I(x,y)} = \{p(x',y') \mid |x'-x| \le r \wedge |y'-y| \le r\}$$

with r representing the L1-distance to the pixel $p(x,y)$. The color value of $p(x,y)$ is represented in the $L^*a^*b^*$ color space and denoted by $\left(L^*_p, A^*_p, B^*_p\right)$. Thus, the mean neighborhood color of $p(x,y)$ in the L^*-band is determined by

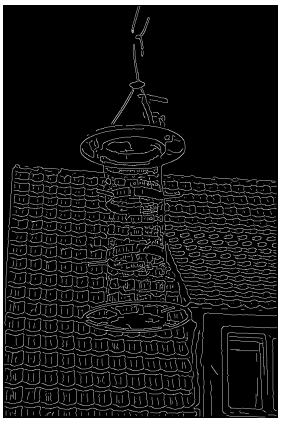
$$L_{\eta^r_{I(x,y)}} = \frac{\sum\limits_{p\in\eta^r_{I(x,y)}} \left(L^*_p\right)}{\left|\eta^r_{I(x,y)}\right|}.$$

The values for the a^*- and b^*-band are denoted by $a_{\eta^r_{I(x,y)}}$ and $b_{\eta^r_{I(x,y)}}$ respectively, so that the mean neighborhood color $Lab_{\eta^r_{I(x,y)}}$ of a pixel $p(x,y)$ is defined by $\left(L_{\eta^r_{I(x,y)}}, a_{\eta^r_{I(x,y)}}, b_{\eta^r_{I(x,y)}}\right)$. According to the International Commission on Illumination *CIE*[1], the color distance $\Delta E^*(u,v)$ between two color values u, v in the $L^*a^*b^*$ color space is calculated by using the Euclidean distance:

$$\Delta E^*(u,v) = \sqrt{\left(L^*_u - L^*_v\right)^2 + \left(a^*_u - a^*_v\right)^2 + \left(b^*_u - b^*_v\right)^2}$$

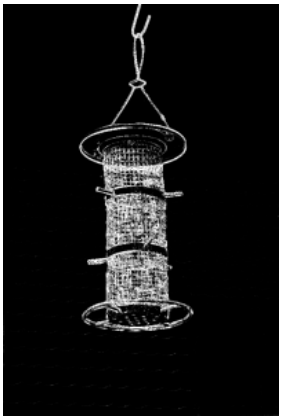
[1]CIE: Commission Internationale de l'eclairage, http://www.cie.co.at

(a) Original image



(b) Canny Edge Detection

(c) Higher Order Statistics



(d) Deviation Scoring

Fig. 3: Comparison of edge detection techniques.

(a) HSV $(0, 1, 1)$ increased in component H.

(b) HSV $(0, 0, 1)$ increased in component S.

(c) HSV $(0, 0, 0)$ increased in component V.

Fig. 4: Visualization of the ΔE^* color distance within each component of the well known HSV color space, with colors $c_0, \ldots, c_9$. Where c_i of the i-th square is increased in each of the components H (Fig. 4a), S (Fig. 4b) and V (Fig. 4c) so that $\Delta E^*(c_i, c_{i+1}) \approx 16$.

For each $p(x, y)$, the neighbor difference $\Delta\eta^r_{(x,y)}$ is then defined by

$$\Delta\eta^r_{I(x,y)} = min\left(255 \cdot \frac{\Delta E^*\left(Lab_{\eta^r_{I(x,y)}}, p\right)}{\Delta E^{max}}, 255\right)$$

with ΔE^{max} being the maximum possible distance in the $L^*a^*b^*$ color space and $\Delta E^*(u, v)$ being the Euclidean distance of the color values u, v in the $L^*a^*b^*$ space. In Fig. 4 we illustrated a color distance of $\Delta E^* = 16$, by varying one of the three components of a base color defined in the HSV color space .

In the following, all pixels $p(x, y)$ with a neighbor difference greater than the threshold $\Theta_{score} \in [0, 255]$ are called *edges* or *edge pixels*, so that the equation $\Delta\eta^r_{I(x,y)} > \Theta_{score}$ holds for each edge pixel of the image. Even though the parameter Θ_{score} could be set freely, we recommend a value of 50 (c.f. Tab. 1) as it showed the best result. Before calculating the score values of the edge pixels, I is convolved using a Gaussian kernel with a standard deviation $\sigma = \Theta_\sigma$ to remove noise and generally soften the image. We recommend to set the value of Θ_σ to $\frac{1}{10}$ (c.f. Tab. 1). The resulting image is then denoted by I'.

Afterwards, another image I'_σ is created by convolving I' once again by using the same Gaussian kernel. $I'$and $I'_\sigma$are then used to compute the score values $\mu \in [0, 255]$ of the edge pixels. Therefore, the score $\mu(x, y)$ for an edge pixel $p(x, y)$ is determined by the squared neighbor difference in the images $I'$and I'_σ at the location of the according pixel:

$$\mu(x, y) = min\left\{255, \left(\Delta\eta^r_{I'(x,y)} - \Delta\eta^r_{I'_\sigma(x,y)}\right)^2\right\}$$

Due to the limitation to $\mu(x, y) \leq 255$, we are treating all color changes between $I'(x, y)$ and $I(x, y)$ equally where $\Delta E > 16$. This can be justified by human perception, which recognizes two colors u, v to as rather unsimilar to each other if $\Delta E^*(u, v) > 12$ [3]. Thus it can be said, that a $\Delta E^* > 16$ indicates a significant color change which is also a strong indication for an edge.

Afterwards, all edge pixels with a score value greater than the threshold Θ_{score} are treated as candidates for the focused region of the image while the score values of all pixels having a score value less than Θ_{score} are set to 0 and are thus no candidates. The resulting candidate set $I_{score}(x, y)$, is defined by the following equation:

$$I_{score}(x, y) = \begin{cases} 0 & \mu(x, y) < \Theta_{score} \\ \mu(x, y) & else \end{cases}$$

An illustration of the candidate set can be seen in Fig. 3d and Fig. 2b, where brighter pixels indicate a large score and black pixels indicate a score less than the threshold Θ_{score}.

3.2 Score Clustering

In this step, clusters are generated from all points in I_{score} in order to find compound regions of focused areas. Therefore, the density-based clustering algorithm DBSCAN [4] is used. In contrast to the K-Means [5], which partitions the image into convex clusters, DBSCAN also supports concave structures which is more desirable in this case. The following section gives a short outline of DBSCAN and then describes how the necessary parameters ε and $minPts$ are determined automatically and how DBSCAN is used in our segmentation algorithm for further processing.

3.2.1 DBSCAN

In this stage, clusters are generated from all $p \in I_{score}$ by applying DBSCAN, which is based on the two parameters ε and $minPts$. The main idea of this clustering algorithm is that each point in a cluster is located in a dense neighborhood of other pixels belonging to the same cluster. The area in which the neighbors must be located is called the *ε-neighborhood* of a point, denoted by $N_\varepsilon(p)$, which is defined as follows:

$$N_\varepsilon(p) = \{q \in D \mid dist(p,q) \leq \varepsilon\} \tag{4}$$

where D is the database of points and $dist(p,q)$ describes the distance measure (e.g. the Euclidean distance) between two points $p, q \in D$.

For each point p of a cluster C there must exist a point $q \in C$ so that p is within the *ε-neighborhood* of q and $N_\varepsilon(p)$ includes at least $minPts$ points. Therefore some definitions are invoked which are described in the following. Considering ε and $minPts$, a point p is called directly *density-reachable* from another point q, if $p \in N_\varepsilon(p)$ and p is a so-called *core-point.* A point p defined as a *core-point* if $|N_\varepsilon(p)| \geq MinPts$ holds. If there exists a chain of n points $p_1, \ldots, p_n$, such that p_{i+1} is *directly density-reachable* from p_i, then p_n is called *density-reachable* from p_1. Two points p and q are *density-connected* if there is a point o from which p and q are both density reachable, considering ε and $minPts$.

Now, a cluster can be defined as a non-empty subset of the Database D, so that for each p and q, the following two conditions hold*:*

- $\forall p, q$: if $p \in C$ and q is *density-reachable* from p, then $q \in C$
- $\forall p, q \in C : p$ is density-connected to q

Points that do not belong to any cluster are treated as $noise = \{p \in D \mid \forall i : p \notin C_i\}$, where $i = 1, \ldots, k$ and $C_1, \ldots, C_k$ are the found clusters in D.

3.2.2 Determination of Parameters

To provide highest flexibility with respect to the different occurrences of the focused area, we do not apply absolute values for ε and $minPts$, but compute them relatively to the size of the image and its score distribution. Thus, ε is calculated by $\varepsilon = \sqrt{|I|} \cdot \Theta_\varepsilon$, with $|I|$ denoting the total amount of pixels of the image represented by I and $\Theta_\varepsilon \in [0,1]$. The second parameter $minPts$ is determined by

$$minPts = \left\lfloor \frac{(\varepsilon+1)^2}{|I|} \left(\sum_{(x,y) \in I_{score}} min\left\{ \frac{\mu(x,y)}{\Theta_{dbscan}}, 1 \right\} \right) \right\rfloor,$$

with the threshold Θ_{dbscan} set to 255.

The result of the DBSCAN clustering is a cluster set $C = \{c_1, \ldots, c_n\}$, with each $c_i \in C$ representing a subset of pixels $p(x,y) \in I$. Due to our assumption that small isolated sharp areas are treated as noise, we define the relevant score cluster set

$$\hat{C} = \left\{ c \in C \mid |c_i| \geq \frac{max_C}{2} \right\} \subseteq C,$$

with $max_C = max\{|c_1|, \ldots, |c_n|\}$ being the amount of pixels of the largest cluster. An illustration of this step can be seen in Fig. 2c on page 5 where different clusters are painted in different colors.

3.3 Mask Approximation

The relevant score cluster set $\hat{C}$, as defined in section 3.2, is already a good reference point of the OOI's location and distribution. In general however, there exists no single contiguous area, but several individual regions of interest representing the focused objects. This stage of the algorithm connects all clusters $c \in \hat{C}$ to a contiguous area which represents an approximate binary mask of the OOI. This is achieved through the two steps *Convex Hull Linking* and *Morphological Filtering,* that will be described in more detail below.

3.3.1 Convex Hull Linking

In the *convex hull linking* step we first generate the convex hull for all points in the ε-neighborhood $N_{Eps}(p)$ of each core point p of the cluster set. Let $K = \{k_1, \ldots, k_j\}$ be the set of all core points from the score clusters in $\hat{C}$ and let $convex(P)$ be the convex hull of a point set P. Then we can define the set of convex hull polygons by $H =$

$\{convex(N_{eps}(k_1)), \ldots, convex(N_{eps}(k_j))\}$ which is used to generate a contiguous area. Therefore each $p(u,v) \in I$ is checked, if it is located within one of the convex hull polygons of H. If that is the case, we mark this pixel with 1, otherwise with 0. The binary approximation mask I_{app} is then given by

$$I_{app}(x,y) = \begin{cases} 1 & if\ \exists H_i : p(x,y) \in H_i \\ 0 & otherwise \end{cases}.$$

Afterwards we apply the morphological filter operations *closing* and *dilation by reconstruction* to I_{app} for smoothing and closing small holes.

3.3.2 Morphological Filtering

Morphological filters are based on the two primary operations *dilation* $\delta_H(I)$ and *erosion* $\varepsilon_H(I)$ where $H(i,j) \in \{0,1\}$ denoting the *structuring element.* For a binary image I, $\delta_H(I)$ and $\varepsilon_H(I)$ are defined as in the following equations:

$$\delta_H(I) = \{(s,t) = (u+i, v+j) \mid (s,t) \in I, (i,j) \in H\}$$

$$\varepsilon_H(I) = \{(s,t) \mid (s+i, t+j) \in I, \forall (i,j) \in H\}$$

The operation *morphological closing* φ_H, is a composition from the two primary operations, so that $\varphi_H(I) = \varepsilon_H(\delta_H(I))$. Thus, the input image I is initially dilated and subsequently eroded, both times with the same *structuring element H.* In order to define the following operation *dilation by reconstruction,* some more definitions are required. At first, the primary operations $\delta_H(I)$ and $\varepsilon_H(I)$ are extended to the *basic geodesic dilation* $\delta^{(1)}(I, I')$ and *basic geodesic erosion* $\varepsilon^{(1)}(I, I')$ of size one as in the following equations.

$$\delta^{(1)}(I, I')(u,v) = min\{\delta_H(I)(u,v), I'(u,v)\}$$

$$\varepsilon^{(1)}(I, I')(u,v) = max\{\varepsilon_H(I)(u,v), I'(u,v)\}$$

Note that these basic geodesic operations need an additional Image I', which is called marker, where the input image I is called mask. Thus, the result of a *geodesic erosion* at position (u,v) is the maximum value of the *erosion* ε_H of mask I and the value of the marker image $I'(u,v)$ and vice versa for the *geodesic dilation.* The geodesic *erosion* $\varepsilon^{(\infty)}$ and *dilation* $\delta^{(\infty)}$ of infinite size, called *reconstruction by erosion* $\varphi^{(rec)}$ and *reconstruction by dilation* $\gamma^{(rec)}$, is then defined as follows

$$\varphi^{(rec)}(I, I') = \varepsilon^{(\infty)}(I, I') = \varepsilon^{(1)} \circ \ldots \circ \varepsilon^{(1)}(I, I')$$

$$\gamma^{(rec)}(I, I') = \delta^{(\infty)}(I, I') = \delta^{(1)} \circ \ldots \circ \delta^{(1)}(I, I')$$

Note that $\varphi^{(rec)}(\cdot,\cdot)$ and $\gamma^{(rec)}(\cdot,\cdot)$ converge and achieve stability after a certain number of iterations. Thus it is assured that these functions do not need to be executed indefinitely and so the application is guaranteed to terminate.

3.3.3 Application

In our approach, we primarily apply a *morphological closing* operation $\varphi_H(I_{app}) = \varepsilon_H(\delta_H(I_{app}))$ to the approximate mask. The dimension of the structuring element H therefore is discussed later. Afterwards we use $\varphi^{rec}(I_{app}, \delta_{H'}(I_{app}))$ to close holes in the approximate mask I_{app}. The dimension of the structuring element H is $h \times h$, where h is calculated relatively to the total pixel count $|I_{app}|$ of the image I_{app}, so that $h = \sqrt{|I_{app}|} \cdot \Theta_{rec}$, with $\Theta_{rec} \in [0,1]$. After this morphological processing, the approximate mask I_{app} covers the OOI quite well (c.f. Fig. 2d on page 5). In general however, it includes boundary regions that exceed the borders of the OOI and tend to surround it with a thick border. The following two stages of our algorithm refine the mask by erasing the surrounding border regions.

3.4 Color Segmentation

In this stage, the pixels from the approximate mask I_{app} are divided into groups, so that each group contains pixels that correspond to similar colors in I. Therefore we process each $p(u,v) \in I_{app}$ and iteratively include all its neighbors n for which the following conditions hold:

$$n \in \left\{(s,t) \in \eta^1_{I_{app}(u,v)} \mid I_{app}(s,t) = I_{app}(u,v) = 1\right\}$$

$$\wedge\ \Delta E^*(p,n) < \Theta_{dist}. \qquad (5)$$

The threshold $\Theta_{dist} \in [0, 100]$ is an internal parameter, which specifies the maximum distance between two color values u, v in the $L^*a^*b^*$ color space.

Therefore a method $expand(x, y, R)$ is called for each $p(x, y) \in \{(x, y) \in I_{app} \mid I_{app}(x, y) = 1\}$, which is not yet marked as visited. $R = \{(x, y)\}$ here defines a new color region formed by the point $p_1(x, y)$. The method $expand(x, y, R)$ then proceeds as follows: For all neighbors $p_2(x, y)$ of $p(x, y)$ fulfilling Eq. 5 on the preceding page we add p_2 to R and mark $p(u, v)$ as visited. Then $expand(u, v, R)$ is called recursively. The resulting set of regions is called R_{color}.

3.5 Region Scoring

In this step, a relevance value μ is calculated for each region $r \in R_{color}$. The more a region is surrounded by areas with a large score μ, the larger the relevancy value gets. Low relevant regions are removed afterwards which causes an update of μ in the neighboring regions and thus possibly triggers another deletion if the relevance of an updated region is not high enough after the according update.

3.5.1 Boundary Overlap

The boundary overlap BO_r^R of a region r is a measure for the adjacency of r to the approximate mask I_{app} and is defined as

$$BO_r^R = |\{(u, v) \in B_r \mid \exists r' \in R : (u, v) \in r'\}|,$$

where B_r is the difference of r to its dilation. The mask boundary overlap MBO_r of r is then defined as $MBO_r = \frac{BO_r^{R_{color}}}{|B_r|}$. MBO_r specifies the ratio of the number of outline points located in other regions to the number of all outline points of r.

The score boundary overlap $SBO_r = \frac{BO_r^{\hat{C}}}{|B_r|}$ of r is a measure for the adjacency of r to the corresponding score values μ. A large SBO_r indicates, that r has a neighborhood with large corresponding score values μ.

3.5.2 Mask Relevance

The mask relevance for a given region r can then be defined as $MR_r = SBO_r \cdot MBO_r$. Afterwards, we eliminate all regions r with a mask relevance value which is too low. The calculation of MR_r is executed iteratively: Let MR_r^i denote the value MR_r of a region r during the i-th iteration. One iteration cycle computes the corresponding μ for each region r and deletes r from the approximate mask I_{app} if $MR_r^i \leq \Theta_{rel}$. The precise assignment of Θ_{rel} and its impact on segmentation quality is discussed later. Once a region r satisfies $MR_r^i \leq \Theta_{rel}$ at iteration i, it will be erased from I_{app}, so that $\forall (x, y) \in r : I_{app}(x, y) = 0$. The calculation of MR_r^i continues for $i = 1, \ldots m$ iterations and terminates as soon as there are no more regions to delete. This is the case, as soon as $MR_r^i = MR_r^{i-1}$ such that $\exists m \geq 1 \mid \forall r \in R_{color} : MR_r^i = MR_r^{i-1}$.

4 Experimental Results

The proposed algorithm is designed to be parameterless and thus applicable to different types of images without having to adjust parameter values by hand. The quality of the resulting segmentation only depends on the size and resolution of the input image. In this section we discuss the quality measure for the comparison of different segmentation algorithms and we show key features, such as the amount of depth of field, that affects difficulties in segmentation. Further more, we demonstrate that images with higher resolution generally lead to be better segmentation results in contrast to the reference algorithms, which loose accuracy with growing size of the processed image.

In 4.3 we describe the internal parameters and threshold values that we determined during our development and testing phases and show their impact on the quality of the segmentation result and the performance.

4.1 Quality measure

To determine the quality of a segmentation mask I we use the *spatial distortion* $d'(I, I_r)$ as proposed in

[7]:

$$d'(I, I_r) = \frac{\sum\limits_{(x,y)} I(x,y) \otimes I_r(x,y)}{\sum\limits_{(x,y)} I_r(x,y)},$$

where $\otimes$ is the binary *XOR* operation and I_r is the manually generated reference mask that represents the ground truth. The spatial distortion denotes the occurred errors, false negatives and false positives, in relation to the size of the reference mask I_r, which is equivalent to the sum of all true positives and false negatives. Notice that d' can grow larger than 1 if more pixels are misclassified than the number of foreground pixels in total. Let ï¿œ be a blank mask so that ï¿œ(x, y)=0 for each pixel $p(x, y)$. The number of false negatives is now equal to the foreground mask pixel in I because no pixel in ï¿œ is defined as forground. For each image I, the spatial distortion d(ï¿œ, I) = 1 . Thus we limit d' to $d \in [0, 1]$, so that $d(I, I_r) = min\{1, d'(I, I_r)\}$.

4.2 Dataset

All experiments were conducted on a diverse dataset of 65 images downloaded from Flickr and created by our own. The images are from different categories with strong variations in the amount of depth of field, as well as in the fuzziness of the background. Also the selection of the images does not focus on certain sceneries, topics or coloring schemes in order to avoid overfitting to certain types of images. In our experiments, we compare the spatial distortion of the proposed algorithm with re-implementations of the works presented in [7], [11] and [18]. The parameters for all algorithms were optimized to achieve the best average spatial distortion over the complete test set.

4.3 Comparison

A major contribution of this algorithm is that none of the parameters introduced in the previous section needs to be hand tuned for an image as all parameters are either independent of the image or determined fully automatically. An overview of the implicit parameters and their default values can be seen in Table 1.

Table 1: Parameters used in the algorithm.

Parameter	Value	Description
Θ_{score}	50	Score μ threshold of I_{score}
Θ_{ϵ}	$\frac{1}{40}$	Spatial radius of DBSCAN
Θ_{σ}	$\frac{1}{10}$	Gaussian blur radius
Θ_{dist}	25	Color similarity distance
Θ_{rec}	$\frac{1}{3}$	Relative size of reconstruction
Θ_{rel}	$\frac{2}{3}$	Minimum value of relevant regions

To calculate the score image I_{score} we use a Gaussian blur with standard deviation of $\Theta_{\sigma} = \frac{1}{10}$. I_{score} is then scaled to fit in 400×400 pixel to improve the processing speed of the subsequent steps without major impact to accuracy. We set $\Theta_{score} = 50$ so that a score μ must exceed 50 to be processed by the density-based clustering algorithm DBSCAN that uses a neighborhood distance $\varepsilon = \Theta_{\epsilon}\sqrt{|I|}$, where $|I|$ is the total pixel count of image I and $minPts$ is calculated in dependence of ε, as described in Sec. 3.2.2. To smooth the approximate mask we use the morphological operation *reconstruction by dilation* γ^{rec} with a structuring element H of size $\sqrt{|I_{approx}|} \cdot \Theta_{rec}$ where $\Theta_{rec} = \frac{1}{3}$. Further we set the maximum distance threshold of similar color values in the $L^*a^*b^*$ color space to $\Theta_{dist} = 25$. The refinement of the approximate mask removes all regions with a mask relevance value less than $\Theta_{rel} = \frac{2}{3}$.

Fig. 6 compares the performance of the reference algorithms with our proposed method. It can be seen that even though the computation time of the proposed algorithm is greater than two of the three reference algorithms, it outperforms the reference algorithms in terms of spatial distortion in all cases. Also, our algorithm has an average spatial distortion error of 0.21 over the complete test set which is less than half compared to the best competing algorithm with an average of 0.51 for the morphological segmentation with region merging [7]. Our algorithm also provides the lowest minimum error of 0.01 in contrast to 0.02 for the fuzzy segmentation algorithm [18].

It should also be noted that in contrast to the reference algorithms, the proposed algorithm shows improved accuracy with larger images whereas the competitors loose accuracy with growing size of the im-

Table 2: Spatial distortion and run time of the proposed algorithm compared to the reference algorithms.

	Proposed	[7]	[18]	[11]
Minimum	**0.01**	0.05	0.02	0.07
Median	**0.10**	0.48	0.93	1
Average	**0.21**	0.51	0.84	0.81
Std.Dev.	**0.21**	0.26	0.25	0.31
Time	28s	9s	54.2s	**2.7s**

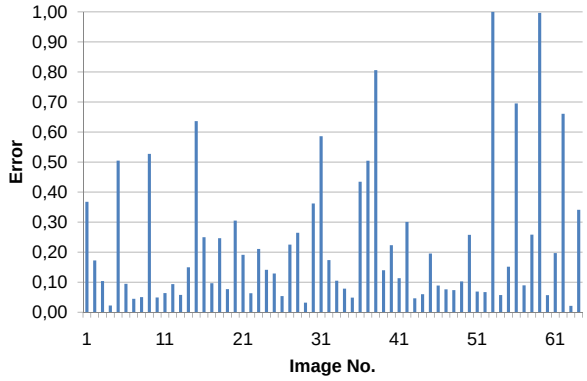


Fig. 5: Spatial distortion error values of the segmentation with our proposed algorithm for each of the 65 dataset images.

age. The minimum, median, average and standard deviation of the spatial distortion error are listed in Tab. 2.

In Fig. 5 we illustrate all spatial distortion error values for each segmented image in the dataset.

4.4 Image Types

For each image we can define key features as in table 3. All these features affect the difficulty for an a accurate segmentation. Images with smaller DOF, homogeneous defocused regions and variant colors tend to produce much better segmentation results than images with high DOF, complex defocused regions and plain colors. The first type of images is commonly used by most of the competitors' publications to ensure a high segmentation quality of the presented algorithm. Figure 7 shows the minor segmentation differences of a small DOF image. A much more challenging task is the segmentation of images with complex background as shown in Fig. 8. Thereby our proposed algorithm achieves a good segmentation result with a spatial distortion of 0.21 (c.f. Fig. 8b), where the other segmentation algorithms [18], [7] and [11] fail with spatial distortion values of each larger than 0.69 (c.f. Fig. 8c, 8d and 8e).

Table 3: Key features of an Image

amount of DOF	defocused regions	color
small	homogeneous	plain
high	complex	variant

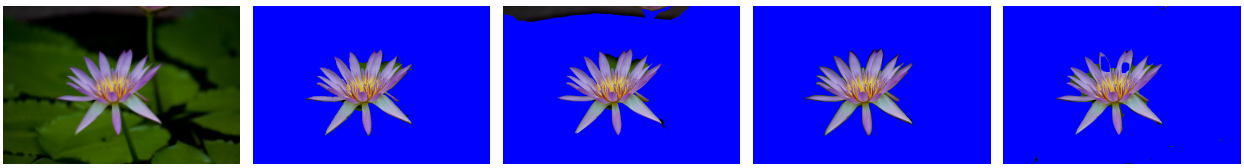
4.5 Size of input image

One of the most influential variables on segmentation quality is the resolution of the input image. A comparatively high resolution is needed for a proper segmentation, if for example an image has just a slightly defocused background and thus shows significant texture. Thus we designed the algorithm to be able to handle a large scope of resolutions properly without loss of quality. By using the image of Fig. 8a as input, a spatial distortion value of 0.65 can already be achieved at the relatively small resolution of 200×300 pixels. For this particular image, a resolution of 240×350 is needed to lower the spatial distortion to 0.42. Fig. 9 shows the diversification of average and standard deviation of the spatial distortion depending on the increase of image size. Because the orientation (portrait or landscape) and aspect ratio (2:3, 4:3, etc.) of the images in our test set varies, we rescale each image so that its longest side equals the value of the resizing operation. In the following we denote the term image size by the longer side of the image.

As we increase the input image size from 100 pixels to 200 pixels, we can lower the average spatial distortion from 0.74 to 0.5 and the standard deviation of the spatial distortion from 0.84 to 0.43. As the size of the images reach 600 pixels, the average and standard deviation of the spatial distortion are 0.36 and 0.25,

(a) Original image. (b) Proposed algorithm. (c) Result of fuzzy segmentation [18] (d) Result of morphological segmentation [7] (e) Result of video segmentation [11]

Fig. 6: Comparing the results of the different algorithms, where the input image has complex defocused regions and small DOF.



(a) Original with homogeneous defocused regions. (b) Proposed algorithm. (c) Result of fuzzy segmentation [18] (d) Result of morphological segmentation [7] (e) Result of video segmentation [11]

Fig. 7: Comparing the results of the different algorithms, where the input image has homogeneous defocused regions, small DOF and variant colors and thus represents an easy task for all algorithms.

(a) (b) (c) (d) (e)

Fig. 8: Results of different segmentation algorithms applied to an image with complex background (Fig. .8a). The spatial distortions of the applied algorithms are 0.21 for our proposed algorithm (Fig. 8b), 1.0 by applying [18] (Fig. 8c), 0.69 by applying [7] (Fig. 8d) and 1.0 by applying [11] (Fig. 8e).

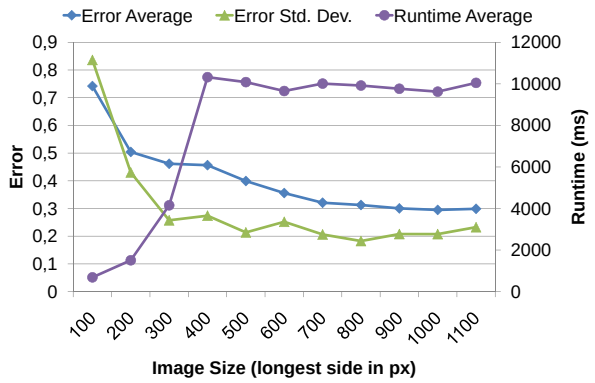


Fig. 9: Impact of the size of the input image to the error rate of the segmentation measured by the spatial distortion and the average runtime per image.

respectively. At the same time, the runtime increases from an average of 686 milliseconds per image at 100 pixels, to 4155 milliseconds at 300 pixels. In our experiments we could also show a significant decrease of the average spatial distortion and the corresponding standard deviation up to an image size of about 600 pixels. For images larger than 800 pixels, the average spatial distortion and standard deviation still improve, yet at a significantly slower rate than in the case of smaller images. In Fig. 9 we summarize the results of this experiment. Because the score image I_{score} was scaled to fit in 400×400 pixels in the score clustering stage, the exponential runtime of the algorithm stagnates after reaching image sizes ≥ 400 pixels for its longest side.

4.6 Sample Segmentations

Fig. 10 presents some segmentation results of different color images. The input image is shown in the first column. In the second column our segmentation result is depicted. The morphological segmentation [7] is placed in the third column, fuzzy segmentation [18] in the fourth column and the video segmentation [11] in the last column. Notice, that in one case the resulting mask covers the entire image (as seen in the second row of Fig. 10 in the third column).

5 Parameter Settings

In this section, we describe the internal parameter settings and threshold values. The choice of the parameter values is the result of an extensive evaluation and represents a recommended set of values that produced the best and stable results. Thus, none of these values has to be set by the user. For each parameter we discuss the impact on the average and standard deviation of the spatial distortion error and on the performance by evaluating the segmentation on our data set.

5.1 Blur Radius

In the first stage, called *deviation scoring*, the parameter Θ_σ determines the size of the radius of the Gaussian convolution kernel to remove noise from the input image. The same kernel is used to re-blur the de-noised image to compute the deviation of the mean neighbor difference of each pixel in the de-noised and re-blurred Image. Fig. 11 shows the distribution of the spatial distortion on the primary y-axis with the mean execution time of the algorithm in milliseconds being displayed on the secondary y-axis. If Θ_σ is set too low, too few noise is removed from the input image. In addition, edges of the OOI in the re-blurred image have a reduced occurrence and thus can be determined less effective in contrast to the edges of the background noise. As illustrated in Fig. 11, a small blur radius of $\Theta_\sigma = 0.775$ results in a high average spatial distortion $d > 0.8$ compared to results of a segmentation with a more convenient choice of Θ_σ. We assign $\Theta_\sigma = 0.9$ as the best trade-off between runtime (approximately 8s per image) and an average spatial distortion of $d_{avg} = 0.26$. Larger values of Θ_σ cause a very intense smoothing operation, so that the edges of the OOI loose their significance.

5.2 Score Clustering Threshold

The score clustering threshold Θ_{score} describes the minimum score value μ, that each pixel has to exceed in order to be processed by the DBSCAN clustering. This parameter ensures that the clustering algorithm does not have to handle a too large number of pixels

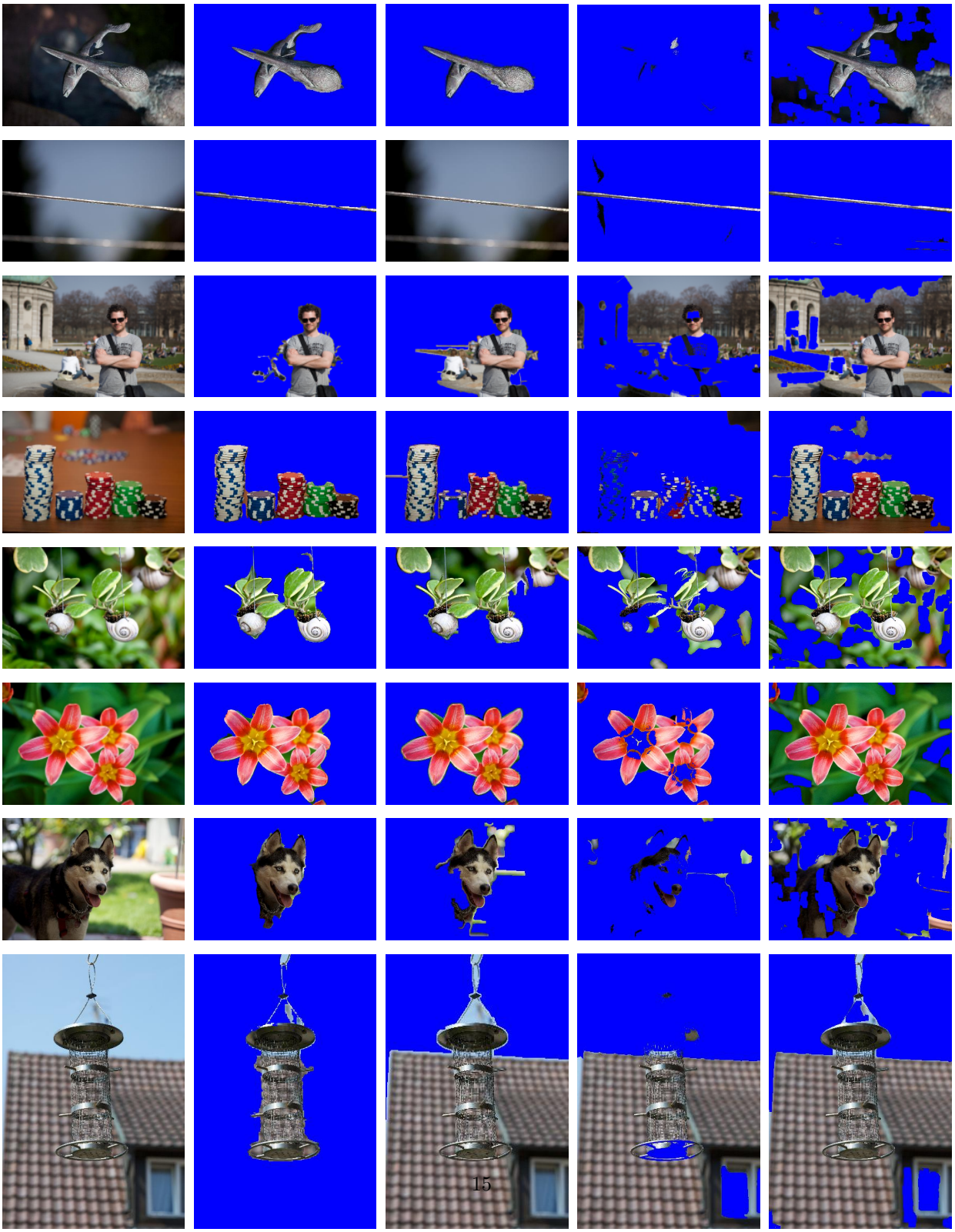


Fig. 10: Segmentation results of different color images (first column) by applying our proposed algorithm (second column), [7] (third column), [18] (fourth column) and [11] (fifth column).

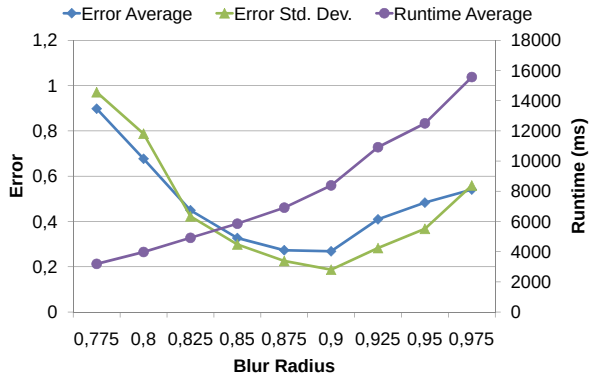


Fig. 11: Impact of the blur radius Θ_σ on the spatial distortion error and average runtime per image.

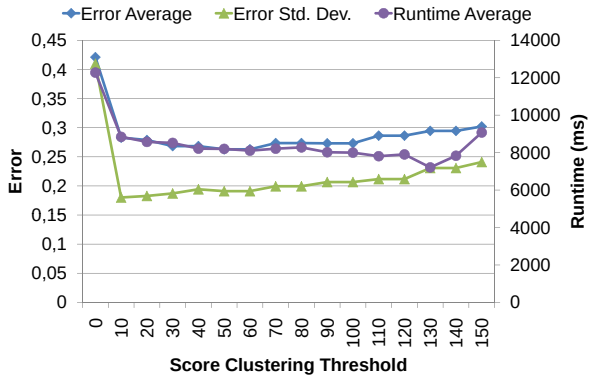


Fig. 12: Impact of the score clustering threshold Θ_{score} on the spatial distortion error and the average runtime per image.

with a score value which is not significant. If Θ_{score} is set too low, DBSCAN will consume more time without improving the segmentation quality significantly. In cases of $\Theta_{score} < 25$ the spatial distortion grows even higher because too many pixels with insignificant score values are considered.

Otherwise if Θ_{score} is set too high, too many pixels with a potentially essential score value are not processed and the remaining clusters contain too few points. This results in a smaller mask that does not cover the whole focus area. As can be seen in Fig. 12, $\Theta_{score} = 50$ can be defined as an optimal trade-off between spatial distortion and runtime.

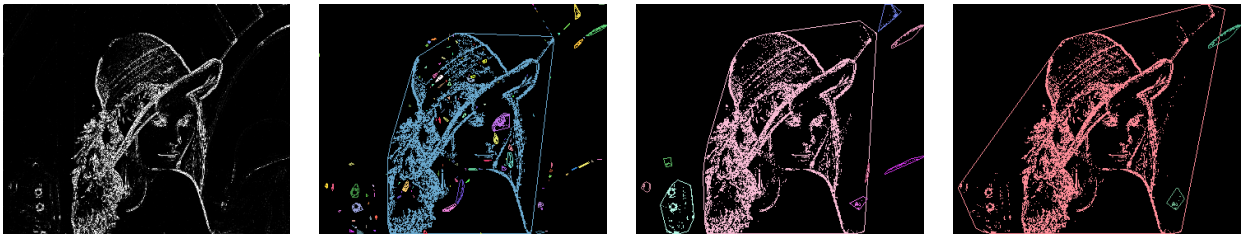
5.3 Neighborhood distance

The neighborhood distance Θ_ε directly influences the size of the ε parameter of DBSCAN, as $\varepsilon = \sqrt{|I|} \cdot \Theta_\varepsilon$. A higher value of Θ_ε increases the spatial radius ε so that a core point has a larger reachability distance. If $minPts$, DBSCAN's second parameter, remains unchanged, an increase in Θ_ε would result in a decreasing number of larger clusters. As $minPts$ is defined relatively to ε (see Sec. 3.2.2) an increase of Θ_ε would also enlarge $minPts$ and vice versa. Fig. 13 illustrates the different clustering results when changing this parameter. If Θ_ε is too low, the main focus area is split into many different, mostly small clusters. Thus, important information in I_{score} would be interpreted as noise, as shown in Fig. 13b.

As you can see in 13c, $\Theta_\varepsilon = 0.025$ produced a much more reliable clustering result and covers the main focus region by not including surrounding noise. If Θ_ε is set too high, too much noise surrounding the OOI is merged with the main cluster (see 13d). The influence of Θ_ε on the segmentation result is shown in figure 14. As optimum distance we set $\Theta_\varepsilon = 0.025$, where the average and standard deviation of the spatial distortion are 0.26 and 0.18, and the average runtime$<$ 9000 milliseconds per image is still acceptable.

5.4 Size of the structuring element

The size of the structuring element H is used in our *mask approximation* stage after the *convex hull linking* step by morphological filter operations, in order to smooth the current mask. If H is too small, little gaps remain and prevent the following *reconstruction by dilation* operation γ^{rec} from filling holes in the approximate mask as only dark regions that are completely surrounded by the white mask are treated as holes. As the size of H increases, more gaps are closed and the contour of the OOIs approximate mask gets more fuzzy. The operation *reconstruction by dilation* γ^{rec} uses a structuring element H'. The size of the structuring element is calculated by $\sqrt{|I|} \cdot \Theta_{rec}$. In



(a) I_{score} from Lena image (b) Clustering result with $\Theta_{\varepsilon} = 0.005$ (c) Clustering result with $\Theta_{\varepsilon} = 0.025$. (d) Clustering result with $\Theta_{\varepsilon} = 0.045$

Fig. 13: Impact of the Θ_{ε} parameter on score clustering. For better visual distinction, each cluster is represented in a random color and surrounded by its convex hull.

(a) (b) (c) (d)

Fig. 15: Impact of the parameter Θ_{rec} on smoothing and filling the approximate mask of the input image (Fig. 15a). Fig. 15b-15d show the approximate masks after *reconstruction by dilation* with small ($\Theta_{rec} = 0.1$), medium ($\Theta_{rec} = 0.25$) and large ($\Theta_{rec} = 0.6$) structuring elements.

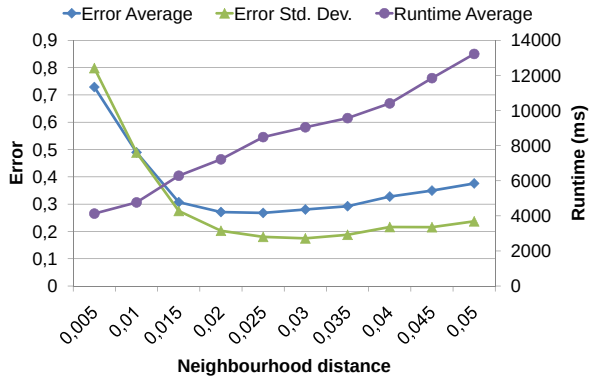


Fig. 14: Impact of the neighborhood distance Θ_ε on the spatial distortion error and the average runtime per image.

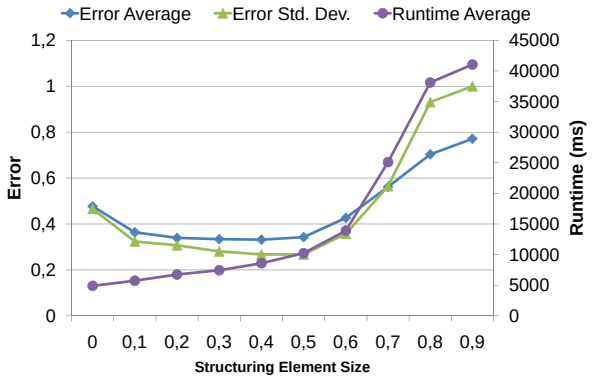


Fig. 16: Impact of the neighborhood distance Θ_{rec} to the spatial distortion error and the average runtime per image.

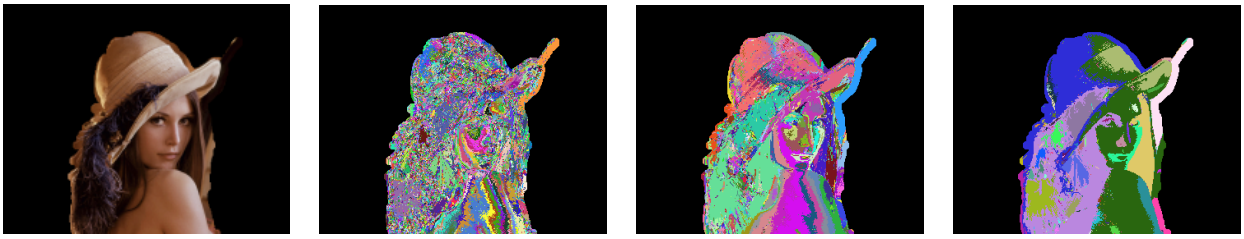
summary it can be said that smaller sizes of H' only cause the filling of small holes in the mask, while larger sizes of H' fill larger holes also. Fig. 15 shows the operation γ^{rec} with small, medium and large structuring elements. For $\Theta_{rec} \in \left[\frac{1}{5}, \frac{1}{2}\right]$ the overall segmentation result of the complete data set is not highly influenced by Θ_{rec} as shown in Fig. 16. Values outside of that range produce a higher error rate. As Θ_{rec} approaches 1, the average runtime also increases rapidly from around $\approx$ 10 seconds at $\Theta_{rec} = \frac{1}{2}$ to $>$ 40 seconds at $\Theta_{rec} = \frac{9}{10}$.

5.5 Color similarity distance

The color similarity parameter Θ_{dist} describes the distance $\Delta E^*(u, v)$ that two colors u, v of the $L^*a^*b^*$ color space may not exceed in order to be considered as similar. Thus, the amount of Θ_{dist} has direct impact on the amount of color regions which are extracted from the original images. If Θ_{dist} is set to a very low value, the resulting color regions are very small. This causes large relative *mask relevance* values in case of small color regions which are overlapped by a small area of the approximation mask. And thus, less regions are removed in the following *region scoring* stage. Too large values for Θ_{dist} imply that too many regions are merged together and thus the *region scoring* will become too vague. Fig. 17 shows the different results of the disjoint color regions by altering the parameter Θ_{dist}. Fig. 17a is the pixel subset of the original image marked by the smoothed mask which is returned by the *mask approximation* stage (Sec. 3.3) of the algorithm. If Θ_{dist} is set to a small value as in Fig. 17b, a large amount of disjoint color regions is created compared to Fig. 17c where $\Theta_{dist} = 25$. Very few color regions are constructed if a large value like $\Theta_{dist} = 40$ is applied to the *color segmentation* stage. This is illustrated in Fig. 17d. Usually, this makes the *region scoring* stage (Sec. 3.5) more vague. The reason for this is that in the case of a large Θ_{dist} it is more likely that $n > 1$ regions $r_1, \ldots, r_n$ with high score variation

$$min\{MR_{r_1}, \ldots, MR_{r_n}\} \ll max\{MR_{r_1}, \ldots, MR_{r_n}\}$$

are merged together in one larger region $r = r_1 \cup \ldots \cup r_n$, where MR_{r_i}, $i = 1 \ldots n$ denotes the *mask relevance* introduced in Sec. 3.5. The deletion of r would then generate more false negatives, which means that the resulting final mask does not cover the entire OOI. The inclusion of r would generate more false positives, so that more background is finally included. As illustrated in Fig. 18, an increasing value of Θ_{dist} also causes an increased processing time. For our tests, we chose $\Theta_{dist} = 25$.



(a) (b) (c) (d)

Fig. 17: Impact of the parameter Θ_{dist} on grouping the smoothed approximation mask into regions of similar colors of the input image (Fig. 17a). Fig. 17b-Fig. 17d show the approximate mask with small ($\Theta_{dist} = 5$), medium ($\Theta_{dist} = 25$) and large (Θ_{dist}=40) values. For better visual distinction, each region is represented in a random color.

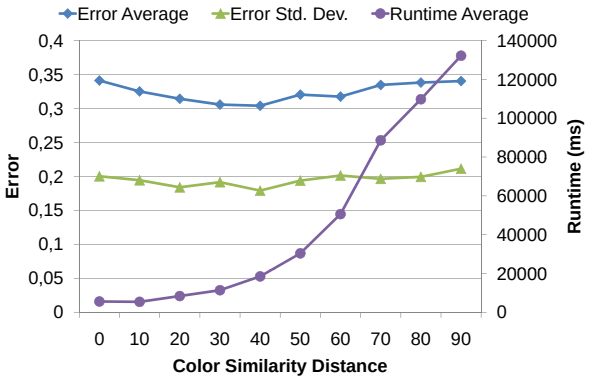


Fig. 18: Impact of the color similarity distance Θ_{dist} on the spatial distortion error and the average runtime per image.

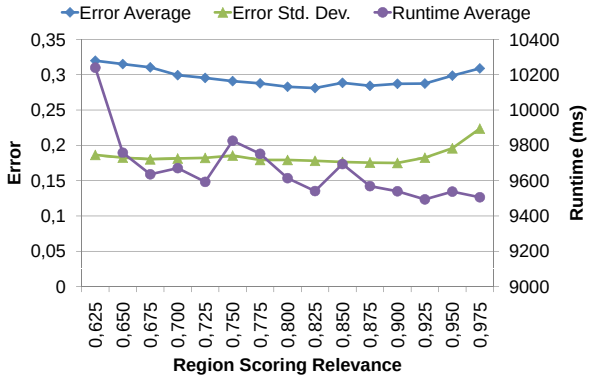


Fig. 19: Impact of the region scoring parameter Θ_{rel} on the spatial distortion error and the average runtime per image.

5.6 Region Scoring Relevance

In the *region scoring* stage we delete each region r from the smoothed approximate mask I_{app} at the i-th iteration if $MR_r^i \leq \Theta_{rel}$. Fig. 19 shows the impact of Θ_{rel} on the segmentation quality. Low values of Θ_{rel} lead to a lower number of deleted regions, leaving the final mask of the OOI surrounded with a thin border in most cases. Thus the total amount of false positives is increased as well as the spatial distortion. The overall influence of Θ_{rel} is rather low because it only affects the refinement step *Region Scoring* (see Sec. 3.5).

6 Impact of DOF on Similarity

Measuring the similarity between two images is an essential step for content-based image retrieval [17] (CBIR). If the image database contains a significant amount of low DOF images, a DOF-based segmentation algorithm can improve the classification accuracy because the extraction of features can be restricted to the subset of pixels contained in the OOI of the images.

Given for example two images I_1 and I_2 which are containing two semantically different objects O_1 and O_2 in their particularly focussed area in front of a comparatively similar background. In this case, the distance between image I_1 and I_2 should be significantly lower than between the extracted Objects O_1 and O_2, such that $d(O_1, O_2) \gg d(I_1, I_2)$.

Fig. 20 shows two sample images that practically display the same scene with a different distance of the lens to the focal plane so that different OOIs are displayed sharply. As our main focus does not lie in the evaluation of best possible feature descriptors or distance measures, we use the well known color histograms for the classification of the data set. For each image we thus create a color histogram h with 12 bins. Between the color histograms $h(I_1)$ and $h(I_2)$ we can now measure the Minkowski-form Distance d_2. For two histograms Q and T, d_p is defined as in Eq. 6 which corresponds to the Euclidean distance in case of $p = 2$.

$$d_p(Q,T) \acute{=} \left(\sum_{i=0}^{N-1} (Q_i - T_i)^p \right)^{\frac{1}{p}} \tag{6}$$

The distance between the color histograms of the complete images $d_2(h(I_1), h(I_2)) = 334$ is considerably lower (3.8 times) than the distance between the histograms of their extracted OOIs $d_2(h(O_1), h(O_2)) = 1277$ (see Fig. 20c and Fig. 20d). This leads to the assumption, that a CBIR system could profit from an automatic segmentation of OOIs in low DOF images to improve search quality.

For a brief verification of this hypothesis, we created a database containing 114 diverse DOF images divided into 17 classes: bird, bee, cat, coke, deer, eagle, airplane, car, fox, apple, ladybird, lion, milk, redtulp, yellowtulp and sunflower.

Let $I_{i,j}$ be the j-th image of the i-th class G_i. We then define the inner-class distance of an image $I_{i,j}$ to be the distance of the image I_{ij} to all other images I_{ik}, $k \neq j$ of the same class G_i

$$dist_{inner}(I_{i,j}) = \frac{1}{|G_i| - 1} \cdot \sum_{J \in G_i / I_{i,j}} d(h(J), h(I_{i,j})),$$

where d is the distance, which is set to the Euclidean Distance d_2 in our case. For a given distance measure d, the average inter-class distance $dist_{inter}$ of an image I_{ij} is the average distance to all other images $J \notin G_i$

$$dist_{inter}(I_{ij}) = \frac{1}{|J \in G_{j \neq i}|} \cdot \sum_{J \in G_{j \neq i}} d(h(J), h(I_{i,j})).$$

In case of a classification task, it is required, that the average inner-class distance is smaller than the inter-class distance of an image. To further improve the classification, it is thus desirable to increase the difference between inter-class and inner-class distance.

In the following experiment, we measured the inner- and inter-class distance for all classes without any segmentation. Afterwards we applied the proposed segmentation algorithm to the images, repeated the experiment and computed the relative changes of the inner- and inter-class distances.

In Fig. 21 we summarize the result of the experiment. It can be seen that the inner-class distance is decreased to an average of 67% of its original value, which is significantly smaller than the decrease of the inter-class distance which decreases to an average of 81% of its original value. Thus, the difference between inter-class and inner-class distance was improved by an average of 14% throughout the data set with a maximum of 27% in the case of the fox class and a minimum of 3% in the case of the (glass of) milk class. These differences between the inner-class distances and the inter-class distances are illustrated in Fig. 22. So it can be said that an CBIR task can profit from the DOF segmentation if the data set contains enough low DOF images.

(a) (b) (c) (d)

Fig. 20: Comparing the results of the different algorithms, where the comparatively similar input images I_1, I_2 (Fig. 20a/20b) have homogeneous defocused regions, small DOF and variant colors. The segmentation results O_1 and O_2 (Fig. 20c / 20d) show rather few similarity.

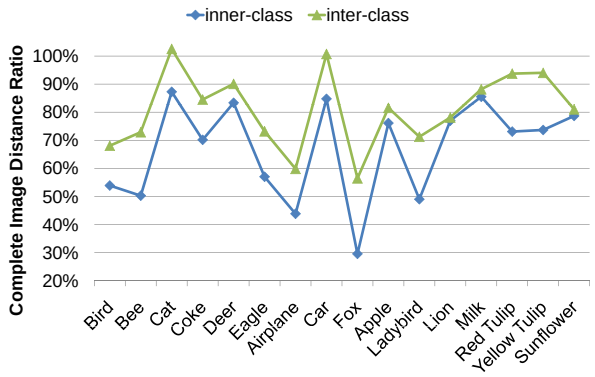


Fig. 21: Impact of the DOF segmentation on the inner- and inter-class distance of a test dataset.

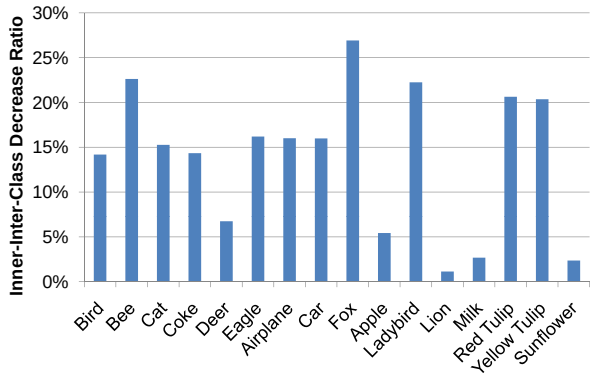


Fig. 22: Amount of percent points that the inner-class distance was lowered more than the inter-class distance.

7 Conclusion

In this paper a new robust algorithm for the segmentation of low DOF images is proposed which does not need to set any parameters by hand as all necessary parameters are determined fully automatically or preset. Experiments are conducted on diverse sets of real world low depth of field images from various categories and the algorithm is compared to three reference algorithms. The experiments show that the algorithm is more robust than the reference algorithms on all tested images and that it performs well even if the DOF is growing larger so that the background begins to show considerable texture. We also demonstrated the positive impact of low DOF segmentation to image similarity in case of CBIR. In our future work, we plan to improve processing speed and accuracy of the algorithm. Furthermore, we plan to apply the algorithm to movies and to apply an automatic detection, whether an image is low DOF or not. A Java WebStart demo of the algorithm can be tested online[2]. We also plan to publish the test data as far as image licensing allows to do so as well as the set including the reference masks for the ROIs. The implementation of the reference algorithms will be made available as ImageJ [1] plugins for download also at the demo URL. We also plan to publish the proposed algorithm as an ImageJ plug-in in the near future.

[2] http://www.dbs.ifi.lmu.de/research/IJCV-ImageSegmentation/

Acknowledgements

This research has been supported in part by the THESEUS program in the CTC and Medico projects. They are funded by the German Federal Ministry of Economics and Technology under the grant number 01MQ
07020. The responsibility for this publication lies with the authors.

We would like to thank Philipp Grosselfinger and Sirithana Tiranardvanich for their ImageJ plug-in implementations of the fuzzy segmentation [18] and video segmentation [11].

References

[1] Abramoff, M., Magelhaes, P., Ram, S.: Image processing with ImageJ. Biophotonics international **11**(7), 36–42 (2004)

[2] Canny, J.: A computational approach to edge detection. Readings in computer vision: issues, problems, principles, and paradigms **184** (1987)

[3] Chung, R., Shimamura, Y.: Quantitative analysis of pictorial color image difference. In: TAGA, pp. 333–345. TAGA; 1998 (2001)

[4] Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD, vol. 96, pp. 226–231 (1996)

[5] Hartigan, J.A.: Clustering Algorithms (1975)

[6] Kavitha, S., Roomi, S., Ramaraj, N.: Lossy compression through segmentation on low depth-of-field images. Digital Signal Processing **19**(1), 59–65 (2009)

[7] Kim, C.: Segmenting a low-depth-of-field image using morphological filters and region merging. IEEE Transactions on Image Processing **14**(10), 1503–1511 (2005)

[8] Kim, C., Hwang, J.: Video object extraction for object-oriented applications. The Journal of VLSI Signal Processing **29**(1), 7–21 (2001)

[9] Kim, C., Park, J., Lee, J., Hwang, J.: Fast extraction of objects of interest from images with low depth of field. ETRI journal **29**(3), 353–362 (2007)

[10] Kovács, L., Szirányi, T.: Focus area extraction by blind deconvolution for defining regions of interest. Pattern Analysis and Machine Intelligence, IEEE Transactions on **29**(6), 1080–1085 (2007)

[11] Li, H., Ngan, K.: Unsupervized video segmentation with low depth of field. IEEE Transactions on Circuits and Systems for Video Technology **17**(12), 1742–1751 (2007)

[12] Li, J., Wang, J., Gray, R., Wiederhold, G.: Multiresolution object-of-interest detection for images with low depth of field. In: Image Analysis and Processing, 1999. Proc. International Conference on, pp. 32–37. IEEE (2002)

[13] Park, J., Kim, C.: Extracting focused object from low depth-of-field image sequences. In: Proceedings of SPIE, vol. 6077, pp. 578–585 (2006)

[14] Tsai, D., Wang, H.: Segmenting focused objects in complex visual images. Pattern Recognition Letters **19**(10), 929–940 (1998)

[15] Won, C., Pyun, K., Gray, R.: Automatic object segmentation in images with low depth of field. In: Image Processing. 2002. Proceedings. 2002 International Conference on, vol. 3, pp. 805–808. IEEE (2002)

[16] Ye, Z., Lu, C.: Unsupervised multiscale focused objects detection using hidden Markov tree. In: Proceedings of the International Conference on Computer Vision, Pattern Recognition, and Image Processing (IEEE Press, 2002), pp. 812–815

[17] Zhang, D., Lu, G.: Evaluation of similarity measurement for image retrieval. In: Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on, vol. 2, pp. 928–931. IEEE (2004)

[18] Zhang, K., Lu, H., Wang, Z., Zhao, Q., Duan, M.: A Fuzzy Segmentation of Salient Region of Interest in Low Depth of Field Image. Advances in Multimedia Modeling pp. 782–791